УДК 004.4
DOI: 10.26102/2310-6018/2023.42.3.022

# Межкомпонентное взаимодействие в мультиагентной системе

**А.В. Бредихин✉, Д.В. Веркошанский, Е.О. Неретин, О.В. Собенина**

*Воронежский государственный технический университет, Воронеж, Российская Федерация*

***Резюме***. Статья посвящена изучению механизмов межкомпонентного взаимодействия в мультиагентных системах. В работе рассмотрены различные подходы к обмену сообщениями между компонентами, а также преимущества и недостатки каждого из них. Определены ключевые проблемы межкомпонентного взаимодействия и предложены их решения. Особое внимание уделено механизму обмена сообщениями на основе брокера сообщений. В статье описаны принципы работы программного брокера, его преимущества и недостатки, а также примеры использования в мультиагентных системах. Результаты исследования показали, что использование брокера сообщений позволяет создать гибкую и масштабируемую систему, способную эффективно обрабатывать большое количество сообщений и поддерживать высокую надежность в работе. В работе представлено описание разработанной структуры формата передачи данных между компонентами мультиагентной системы. Показаны схемы маршрутизации сообщений в рамках системы с использованием брокера сообщений. Описана настройка для реализации разработанных схем межкомпонентного взаимодействия. Предложен механизм кодирования сообщений на основе тэг-ключей, который позволяет проводить их идентификацию для дальнейшей обработки программными агентами. Этот подход может быть полезен при проектировании и разработке различных мультиагентных систем, где необходим обмен сообщениями между различными программными агентами.

***Ключевые слова:*** мультиагентная система, брокер сообщений, формат данных, JSON, RabbitMQ, MAS, кодификация.

***Для цитирования:*** Бредихин А.В., Веркошанский Д.В., Неретин Е.О., Собенина О.В. Межкомпонентное взаимодействие в мультиагентной системе. *Моделирование, оптимизация и информационные технологии.* 2023;11(3). URL: https://moitvivt.ru/ru/journal/pdf?id=1407 DOI: 10.26102/2310-6018/2023.42.3.022

# Intercomponent interaction in a multi-agent system

**A.V. Bredikhin✉, D.V. Verkoshansky, E.O. Neretin, O.V. Sobenina**

*Voronezh State Technical University, Voronezh, the Russian Federation*

***Abstract.*** The article examines the mechanisms of intercomponent interaction in multi-agent systems. The paper discusses various approaches to messaging between components as well as the advantages and disadvantages of each of them. The key problems of intercomponent interaction are identified and their solutions are proposed. Particular attention is paid to the messaging mechanism based on the message broker. The principles of the broker, its advantages and disadvantages as well as examples of use in multi-agent systems are described. The results of the study showed that the use of the message broker makes it possible to create a flexible and scalable system that can efficiently process a large number of messages and maintain high reliability in operation. The paper presents a description of the data transfer format structure between the components of a multi-agent system. Message routing schemes within the system using a message broker are shown. The configuration for the implementation of the intercomponent interaction schemes is described. A mechanism for encoding messages based on tag keys is proposed, which enables their identification for further processing by software agents. This approach can be useful in the design and development of various multi-agent systems, where it is necessary to exchange messages between different software agents.

 

***Keywords:*** multi-agent system, message broker, data format, JSON, RabbitMQ, MAS, coding.

***For citation:*** Bredikhin A.V., Verkoshansky D.V., Neretin E.O., Sobenina O.V. Intercomponent interaction in a multi-agent system. *Modeling, Optimization and Information Technology*. 2023;11(3). URL: https://moitvivt.ru/ru/journal/pdf?id=1407 DOI: 10.26102/2310-6018/2023.42.3.022 (In Russ.).

## Введение

Мультиагентное взаимодействие относится к взаимодействию между несколькими автономными агентами, которые способны воспринимать окружающую среду, рассуждать и принимать решения на основе своих целей и задач. Этот тип взаимодействия может происходить в самых разных условиях, включая компьютерные сети, робототехнику и социальные сети. Одной из главных проблем при многоагентном взаимодействии является координация действий нескольких агентов для достижения общей цели.

Внедрение мультиагентного взаимодействия в систему планирования производства предприятия может быть уместным в определенных ситуациях. В целом, решение об использовании мультиагентных систем (MAS) в корпоративной системе планирования производства должно основываться на конкретных потребностях предприятия. Одним из потенциальных преимуществ использования MAS в планировании производства является возможность моделирования сложных систем с участием множества заинтересованных сторон, таких как ресурсы, заказчики и исполнители. Используя MAS, возможно, удастся добиться большей координации и сотрудничества между этими сторонами, что приведет к повышению эффективности и оперативности реагирования в производственном процессе. Однако внедрение MAS также может быть сложной задачей и может потребовать значительных изменений в существующей инфраструктуре, процессах и культуре организации. Кроме того, сложность MAS может затруднить разработку и внедрение эффективных механизмов контроля, которые могут потребоваться для предотвращения непреднамеренного аварийного поведения [1].

В целом, хотя внедрение мультиагентной системы в систему планирования производства предприятия достаточно трудоемко, оно может принести значительные преимущества.

Улучшенная координация: мультиагентные системы могут улучшить координацию между различными агентами и заинтересованными сторонами, вовлеченными в производственный процесс, что может привести к повышению эффективности и оперативности реагирования.

Гибкость: мультиагентные системы могут быть легко адаптируемыми и гибкими, что облегчает адаптацию к изменяющимся условиям и требованиям производственного процесса.

Масштабируемость: мультиагентные системы могут быть расширены или уменьшены в соответствии с потребностями производственного процесса, что облегчает обработку изменений спроса или производственных мощностей.

Распределенная обработка: мультиагентные системы могут распределять обработку между несколькими агентами, что может повысить производительность системы и снизить риск системных сбоев.

Инновации: мультиагентные системы могут способствовать инновациям и экспериментам, позволяя агентам взаимодействовать и учиться друг у друга, что приводит к появлению новых идей и подходов к планированию производства.

Преимущества внедрения системы мультиагентного взаимодействия в систему планирования производства предприятия могут перевесить недостатки при условии

надлежащего управления рисками и тщательного проектирования, и внедрения системы. Надлежащее планирование, тестирование и мониторинг могут помочь обеспечить успех системы и позволить организациям воспользоваться преимуществами такого подхода [2].

## Материалы и методы

Эффективное межкомпонентное взаимодействие необходимо для создания модульных и масштабируемых программных систем, которые могут адаптироваться к меняющимся требованиям с течением времени. Примеры межкомпонентного взаимодействия в программных системах включают связь между различными частями системы, связь между различными службами в архитектуре микросервисов и связь между различными модулями в системе на основе компонентов.

Например, система AnyLogic предоставляет возможности моделирования для имитации межкомпонентных взаимодействий. В любой логике компоненты могут быть созданы как отдельные агенты с определенными свойствами и поведением. Эти агенты могут взаимодействовать друг с другом и окружающей средой через различные каналы связи, такие как передача сообщений [3].

Управление системой с большим количеством компонентов может быть сложной задачей, поскольку для этого требуется значительное количество усилий и ресурсов. Некоторые трудности, которые могут возникнуть, включают:

1. Сложность: большое количество компонентов может привести к созданию очень сложной системы, которой трудно управлять. Определить причину проблемы или определить, как изменение в одном компоненте может повлиять на другие компоненты, может оказаться непросто.

2. Масштабируемость: по мере увеличения количества компонентов в системе может стать сложнее масштабировать систему для обработки возросшего трафика или пользовательского спроса.

3. Координация: при наличии большого количества компонентов в системе может быть трудно координировать действия разных команд, ответственных за разные компоненты. Это может привести к сбоям в коммуникации, дублированию усилий или конфликту приоритетов.

4. Производительность: Систему с большим количеством компонентов может быть сложнее оптимизировать с точки зрения производительности. Может оказаться непросто выявить узкие места или оптимизировать систему без негативного воздействия на другие компоненты.

5. Техническое обслуживание: при большом количестве компонентов техническое обслуживание может стать более сложным и отнимающим много времени, особенно если компоненты распределены по разным системам и местоположениям.

6. Тестирование и отладка: Тестирование и отладка большой системы может быть сложной задачей, поскольку необходимо учитывать множество различных компонентов и взаимодействий. Это может привести к более длительным циклам разработки и задержкам в выпуске новых функций или исправлении ошибок.

Для модерирования такой системы важно иметь четкое представление об архитектуре системы и взаимодействии между компонентами.

Концепция разрабатываемой платформы подразумевает наличие в сети огромного количества компонентов, большую часть из которых составляют программные агенты, имеющие единый программный код, но разные поведенческие паттерны. При определении способа их взаимодействия необходимо отталкиваться с

учетом онтологии построения системы. На Рисунке 1 представлена схема взаимодействия ряда компонентов.

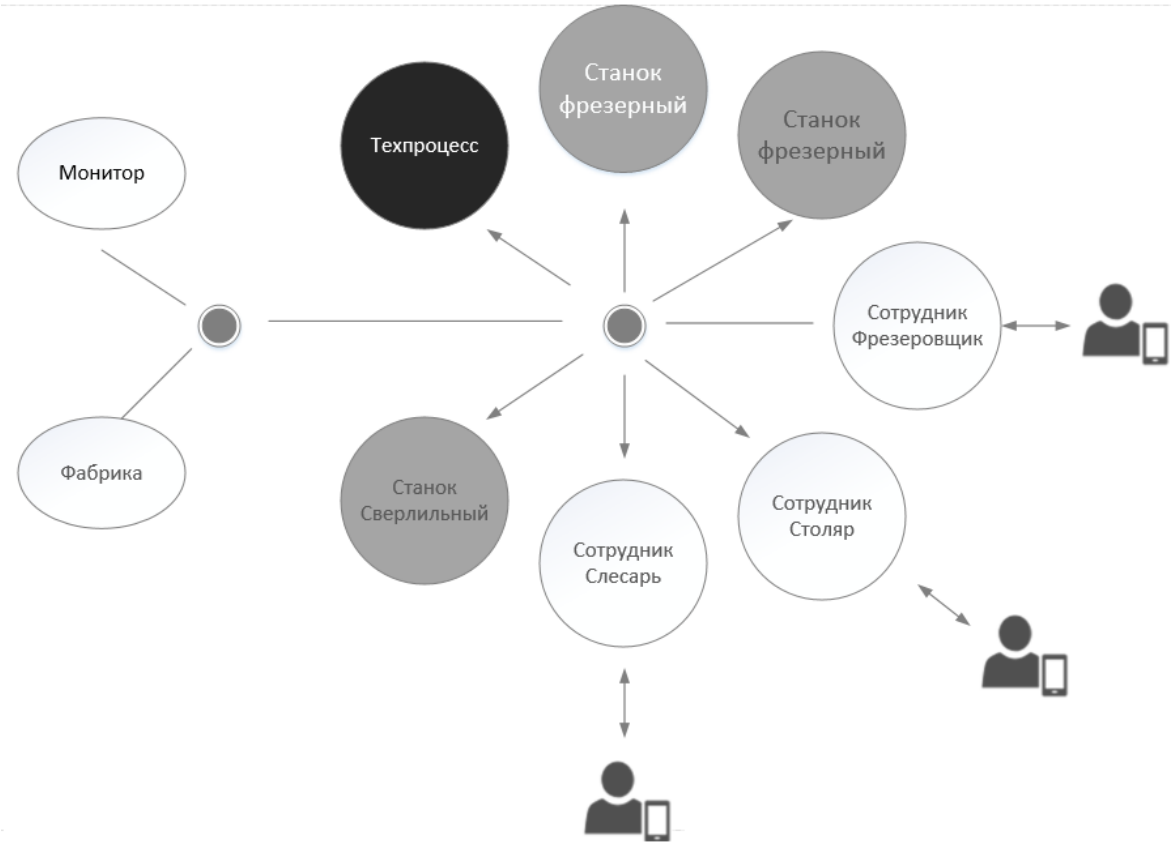


Рисунок 1 – Схема взаимодействия компонентов
Figure 1 – Component interaction scheme

Централизованная система мониторинга может помочь быстро выявить проблемы и обеспечить эффективную координацию между командами, ответственными за различные компоненты. Автоматическое тестирование и развертывание также могут помочь гарантировать, что изменения в одном компоненте не окажут негативного влияния на систему в целом [4].

Подключение компонентов с использованием JSON, API и RabbitMQ может стать хорошей отправной точкой для решения проблемы объединения компонентов в системе. Так, брокер сообщений, имеющий значительную пропускную способность, позволяет тысячам компонентов системы взаимодействовать между собой одновременно, а стандартизация сообщений с использованием JSON позволит универсализировать методы взаимодействия различных компонентов [5].

## Результаты

Проектирование формата JSON было решено начать с выделения универсальной информации, передаваемой в любом типе сообщений:

1. Тип сообщения – позволяет определить назначение сообщения, для его дальнейшей обработки.

2. Отправитель и получатель – позволяет не только проверить, являетесь ли вы получателем данного сообщения, но производить общий мониторинг отправляемых и получаемых сообщений системы.

3. Параметры – содержащие все не универсальные данные, передаваемые с сообщением.

Дальнейшая структура файла хоть и имеет схожие части, но большое количество разнообразных компонентов системы не позволяет выделить их, вследствие чего дальнейшую структуру рассмотрим на примере сообщений, передаваемых между агентами системы.

Так, одними из наиболее часто встречаемых сообщений являются запрос и ответ на выполнения работы, не имеющие структурных различий.

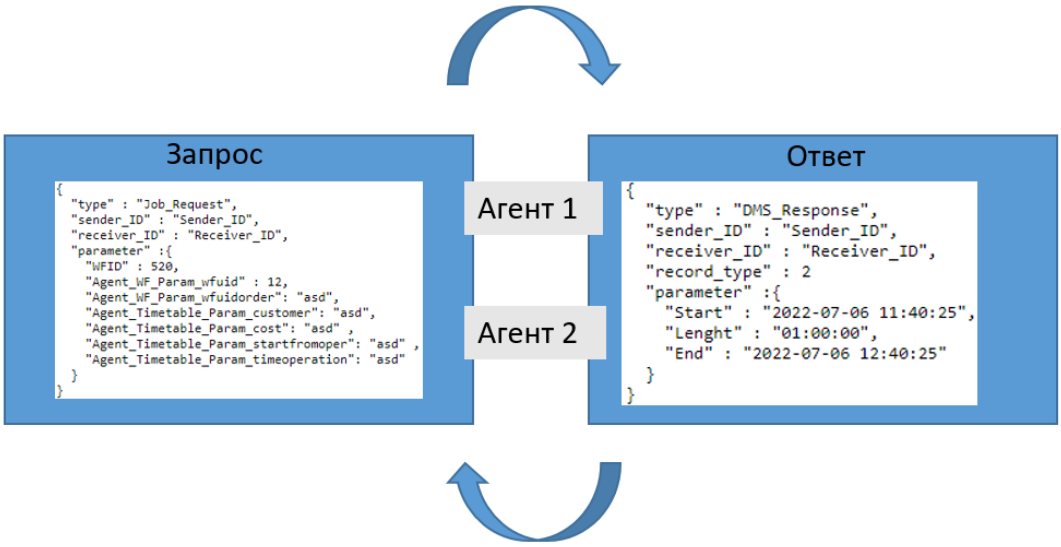


Рисунок 2 – Пример структуры сообщения
Figure 2 – Message structure sample

Идентифицируя тип сообщения, используя соответствующий параметр, сообщение обрабатывается необходимым способом, для продолжения работы программы. Соответсвующий набор реакций в каждом компоненте системы и будет являться программным интерфейсом, позволяющим обеспечить взаимодействия различных частей системы.

Разрабатывая принцип маршрутизации для данной системы, можно отметить следующую специфическую проблему. Каждый агент не имеет представления о количестве и специализации других агентов. Решить данную проблему можно с помощью шаблона и функции обмена сообщениями «Topics», которые обеспечивают гибкую и точную маршрутизацию сообщений на основе соответствующих критериев. Темы предоставляют способ выборочного распространения сообщений среди нескольких потребителей на основе содержимого сообщения. Функция «Topics» в RabbitMQ реализована с использованием концепции «topic exchanges». Topic exchanges получает сообщения от производителей и направляет их в одну или несколько очередей на основе ключа маршрутизации сообщения. Ключ маршрутизации – это строка, связанная с каждым сообщением. Когда сообщение публикуется в topic exchanges, брокер оценивает ключ маршрутизации сообщения по шаблонам маршрутизации, заданным привязками очередей. Ключ маршрутизации обычно представляет собой строку, разделенную точками, где каждый сегмент представляет «тему» маршрутизации [6].

Темы поддерживают два специальных знака:

– звездочка (*): соответствует одному слову в сегменте ключа маршрутизации. Например, привязка ключа "Агент.*". будет соответствовать «Агент.Станок», но не «Агент.Станок.Сверлильный»;

– знак решетка (#): соответствует нулю или более словам в сегменте ключа маршрутизации. Его можно использовать только в конце ключа маршрутизации. Например, ключ привязки «Агент.*» будет соответствовать «Агент.Станок» и «Агент.Станок.Сверлильный».

Использование этих специальных знаков и шаблонов ключей маршрутизации, «Topics» обеспечивают гибкую и мощную маршрутизацию сообщений. Они позволяют динамически и выборочно потреблять сообщения на основе определенных критериев. Темы особенно полезны в сценариях, где необходимо рассылать сообщения нескольким потребителям на основе разных категорий или тем.

Построим схему маршрутизации на примере сообщений при рассылке «запросов на работу» от техпроцесса к исполнителям. Присвоим агентам их очереди в соответствии с их специализацией и номером. Затем, используя специальные знаки, зададим ключи очереди каждого агента. Получим два вида ключей:

– «Агент.1.#» – позволяющий принимать сообщения, направленные конкретному агенту вне зависимости от его специализации;

– «#.Фрезерный» – позволяющий получать сообщения, направленные всему фрезерному оборудованию.

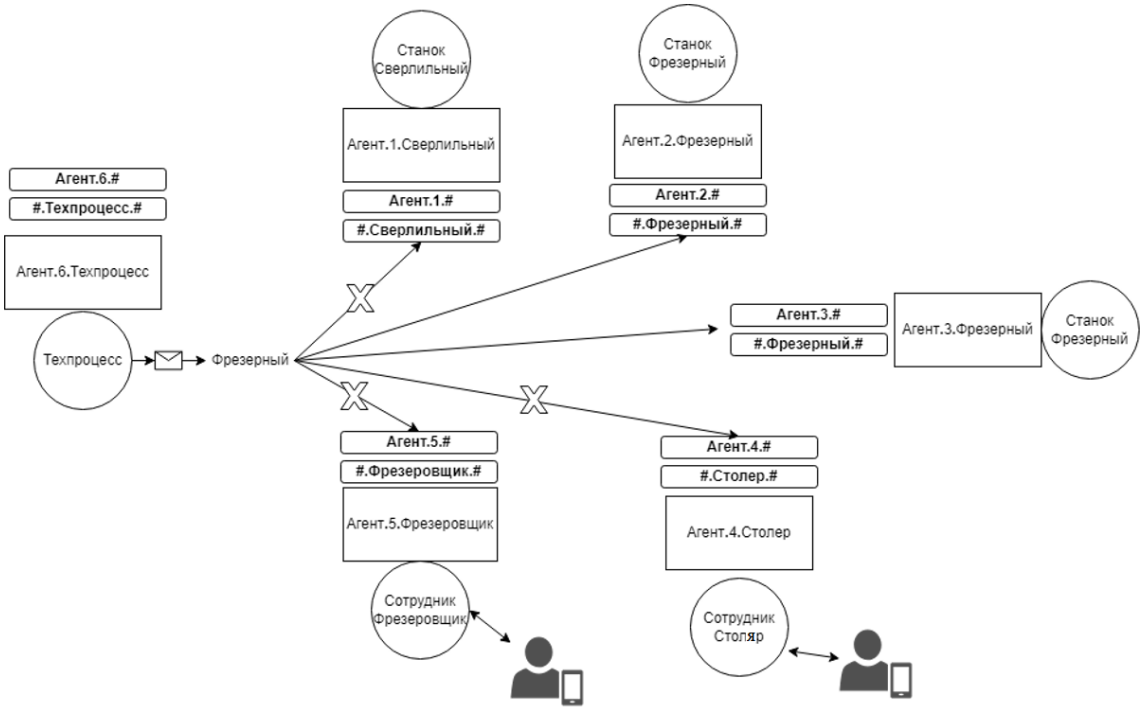


Рисунок 3 – Схема организации передачи сообщения
Figure 3 – Message transfer organization scheme

Так, на Рисунке 3 показано, что сообщение, отправленное техпроцессом, получат лишь 2 агента, соответствующие теме сообщения, личный же ключ сообщений у техпроцесса позволит послать ответ конкретно ему.

Задачи адресации и маршрутизации берет на себя брокер сообщений RabbitMQ, позволяющий гибко настраивать и регистрировать поток всех сообщений, циркулирующих в системе [7, 8].

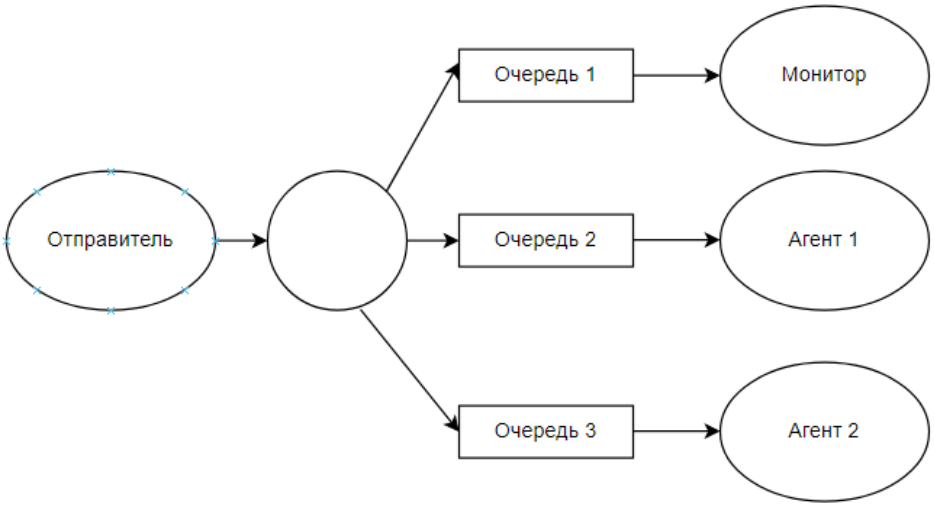


Рисунок 4 – Организация очередей сообщений брокера
Figure 4 – Broker message queuing

Routing key – ключи навигации, по сути являющиеся аналогом почтового адреса, не являются уникальными конкретной очереди сообщений. Это позволяет как отправлять сообщение нескольким получателям сразу, так и посылать личные сообщения.

Процесс подбора исполнителя для предстоящей операции предполагает постоянное взаимодействие различных ресурсов между собой, с целью определения наиболее выгодного для операции решения. Чтобы обеспечить синхронность действий, было решено использовать ключи сообщений в виде кодов, позволяющих определить назначение сообщения, и этап подбора.

Код сообщения представляет собой набор чисел, разделенных дефисом, и формируется следующим образом:

– первое число отвечает за тип сообщения (Таблица 1);
– второе число отвечает за то, от кого отправлено сообщение (Таблица 2);
– третье число символизирует получателя (Таблица 2);
– четвертое число символизирует назначение сообщения.

Таблица 1 – Типы сообщений
Table 1 – Message types

| Код | Описание |
|---|---|
| 0 | Запрос |
| 1 | Ответ |
| 2 | Оповещение |
| 3 | Сообщение |

Таблица 2 – Отправители или получатели
Table 2 – Senders and recipients

| Код | Описание |
|---|---|
| 0 | Агент техпроцесс |
| 1 | Агент ресурс |
| 2 | Субагент |
| 3 | Монитор |
| 4 | Сервис принятия решений |
| 5 | Мобильное приложение |
| 6 | Фабрика агентов |

После согласования правил кодификации сообщений были выделены следующие типы сообщений, необходимые для межагентного взаимодействия (Таблица 3):

– запрос на работу – сообщение, отправляемое агентом-техпроцессом всем агентам-ресурсам, подходящим для выполнения данной операции. Агент-ресурс получает сообщение с необходимыми параметрами. Реакцией на данное сообщение является запуск нового экземпляра Workflow процесса «Запрос на работу»;

– запрос сервису принятия решения – сообщение, отправляемое агентом-ресурсом в сервис принятия решения, для определения наиболее выгодного решения для агента. Сервис принятия решения получает информацию о потенциальной работе и текущее расписание агента-ресурса. Реакцией на данное сообщение будет расчет актуальности потенциальной операции и соответствующий ответ;

– ответ сервиса принятия решения – сообщение, отправляемое в ответ на запрос агента-ресурса, содержащее результат вычислений. Реакцией на данное сообщение является продолжение ранее запущенного Workflow процесса «запрос на работу»;

– ответ на работу – сообщение, отправляемое агентом-ресурсом на агент-техпроцесс, с целью подтвердить свое участие в предстоящей операции. Параметрами, передаваемыми с данным сообщением, являются возможные даты начала и конца операции, а также ее стоимость. Реакцией на данное сообщение является сбор кандидатов на выполнение предстоящей операции;

– оповещение о подтверждении – сообщение, отправляемое агентом-техпроцессом на агент-ресурс, с утверждением его в качестве исполнителя предстоящей операции.

Таблица 3 – Отправители или получатели
Table 3 – Interagent messages

| Код события | Наименование события |
|---|---|
| 0-0-1-1 | Запрос на работу |
| 0-1-4-1 | Запрос для СПР |
| 1-4-1-1 | Ответ СПР |
| 1-1-0-1 | Ответ на работу |
| 3-0-1-1 | Оповещение о подтверждении |

## Обсуждение

Предложенные решения были апробированы в рамках прототипа мультиагентной системы, что позволило выделить ряд преимуществ и недостатков.

К преимуществам можно отнести следующее:

1. Гибкость и масштабируемость: RabbitMQ является гибким и масштабируемым механизмом для организации обмена сообщениями между компонентами. Он может

обрабатывать большое количество сообщений и поддерживать высокую надежность в работе.

2. Брокер сообщений гарантирует надежность доставки сообщений и обеспечивает высокий уровень надежности в работе. Он может обрабатывать ошибки и перенаправлять сообщения в случае сбоев в системе.

3. Управление потоками, что обеспечивает контроль над потоками сообщений и позволяет легко управлять их передачей.

4. Наличие широкого набора API позволяет реализовать программную адаптацию и работу с различными языками программирования, платформами и протоколами.

5. Поддержка протоколов AMQP, MQTT, STOMP может использоваться в различных системах.

Недостатки использования брокеров сообщений в рамках мультиагентной системы:

1. Необходимость установки и настройки: используемый брокер RabbitMQ требует установки и настройки.

3. RabbitMQ может создавать дополнительные накладные расходы на производительность и потребление ресурсов.

4. RabbitMQ требует обслуживания и мониторинга, чтобы обеспечить его правильную работу [9, 10].

В целом, RabbitMQ является мощным и гибким механизмом для организации межкомпонентного взаимодействия, который обладает рядом преимуществ и недостатков. При выборе RabbitMQ для организации межкомпонентного взаимодействия требуется учитывать особенности проекта.

## Заключение

Таким образом, на основе проведенного исследования был реализован механизм обмена сообщениями программными агентами в мультиагентной системе на основе брокера сообщений Rabbitmq. Были исследованы преимущества и недостатки данного подхода, а также проведено сравнение с другими подходами к обмену сообщениями в мультиагентных системах. Результаты показали, что использование брокера сообщений Rabbitmq позволяет создать гибкую и масштабируемую систему, способную эффективно обрабатывать большое количество сообщений и поддерживать высокую надежность в работе. Этот подход может быть полезен при проектировании и разработке различных мультиагентных систем, где необходим обмен сообщениями между различными агентами.

## СПИСОК ИСТОЧНИКОВ

1. Юлейси Г.П., Холод И.И. Взаимодействие в многоагентных системах интеллектуального анализа данных. *Известия СПбГЭТУ ЛЭТИ*. 2020;3:18–23.
2. Скобелев П.О., Иващенко А.В., Андреев М.В., Бабанин И.О. Мультиагентные технологии для управления распределением производственных ресурсов в реальном времени. *Механика, управление и информатика*. 2011;5:110–122.
3. Тимонин А.Н. Обзор инструмента имитационного моделирования Anylogic. *Информационные технологии в образовании*. 2021;4:231–237.
4. Лихтенштейн В.Е., Конявский В.А., Росс Г.В., Лось В.П. *Мультиагентные системы: самоорганизация и развитие*. М.: Финансы и статистика; 2018. 264 с.
5. Sabir B.E., Youssfi M., Bouttane O., Allali H. Authentication and load balancing scheme based on JSON Token for Multi-Agent Systems. *Procedia computer science*. 2019;148:562–570. DOI: 10.1016/j.procs.2019.01.029.

6. Горбунов В.В. Расширенные способы взаимодействия с сервисной шиной предприятия. *Инновации. Наука. Образование*. 2020;23:152–158.
7. Ayanoglu E., Aytas Y., Nahum D. *Mastering RabbitMQ*. Packt Publishing Ltd; 2016. 286 p.
8. Кавалерова А.С., Самочадин А.В., Тимофеев Д.А. Модуль авторизации подписчиков в системе управления очередями rabbitmq. *Неделя науки СПбПУ, 19–24 ноября 2018, Санкт-Петербург.* СПб.: Политех-Пресс; 2019. С. 177–180.
9. Dobbelaere P., Esmaili K.S. Kafka versus RabbitMQ: A comparative study of two industry reference publish/subscribe implementations: Industry Paper. *Proceedings of the 11th ACM international conference on distributed and event-based systems, June 2017*. 2017. p. 227–238. DOI: 10.1145/3093742.3093908.
10. Vandikas K., Tsiatsis V. Performance evaluation of an IoT platform. *2014 Eighth International Conference on Next Generation Mobile Apps, Services and Technologies*. *IEEE, Oxford, UK*. 2014. p. 141–146. DOI: 10.1109/NGMAST.2014.66.

## REFERENCES

1. Yuleisi G.P., Kholod I.I. Interaction in multi-agent data mining systems. *Izvestiya SPbGETU LETI = Proceedings of Saint Petersburg Electrotechnical University*. 2020;3:18–23. (In Russ.).
2. Skobelev P.O., Ivashchenko A.V., Andreev M.V., Babanin I.O. Multi-agent technologies for managing the distribution of production resources in real time. *Mekhanika, upravlenie i informatika*. 2011;5:110–122. (In Russ.).
3. Timonin A.N. Overview of the Anylogic simulation tool. *Informatsionnye tekhnologii v obrazovanii= Information Technology in Education.* 2021;4:231–237. (In Russ.).
4. Liechtenstein V.E., Konyavskii V.A., Ross G.V., Los' V.P. *Multi-agent systems: self-organization and development*. Moscow, Finansy i Statistica; 2018. 264 p. (In Russ.).
5. Sabir B.E., Youssfi M., Bouttane O., Allali H. Authentication and load balancing scheme based on JSON Token for Multi-Agent Systems. *Procedia computer science*. 2019;148:562–570. DOI: 10.1016/j.procs.2019.01.029.
6. Gorbunov V.V. Expanded methods of interaction with the service bus of the enterprise. *Innovatsii. Nauka. Obrazovanie*. 2020;23:152–158. (In Russ.).
7. Ayanoglu E., Aytas Y., Nahum D. *Mastering RabbitMQ*. Packt Publishing Ltd; 2016. 286 p.
8. Kavalerova A.S., Samochadin A.V., Timofeev D.A. Subscriber authorization module in the rabbitmq queue management system. *SPbPU Science Week, 19–24 November 2018, Saint Petersburg.* Saint Petersburg, Politekh-Press; 2019. p. 177–180. (In Russ.).
9. Dobbelaere P., Esmaili K.S. Kafka versus RabbitMQ: A comparative study of two industry reference publish/subscribe implementations: Industry Paper. *Proceedings of the 11th ACM international conference on distributed and event-based systems, June 2017*. 2017. p. 227–238. DOI: 10.1145/3093742.3093908.
10. Vandikas K., Tsiatsis V. Performance evaluation of an IoT platform. *2014 Eighth International Conference on Next Generation Mobile Apps, Services and Technologies*. *IEEE, Oxford, UK*. 2014. p. 141–146. DOI: 10.1109/NGMAST.2014.66.

## ИНФОРМАЦИЯ ОБ АВТОРАХ / INFORMATION ABOUT THE AUTHORS

**Бредихин Алексей Вячеславович,** кандидат технических наук, доцент Воронежского государственного технического университета, Воронеж, Российская Федерация.
*e-mail*: maypochta@yandex.ru
ORCID: 0009-0000-7990-5280

**Aleksey Vyacheslavovich Bredikhin,** Candidate of Technical Sciences, Associate Professor, Voronezh State Technical University, Voronezh, the Russian Federation.

**Веркошанский Денис Валерьевич,** студент Воронежского государственного технического университета, Воронеж, Российская Федерация.

**Denis Valerievich Verkoshansky,** Undergraduate Student, Voronezh State Technical University, Voronezh, the Russian Federation.

**Неретин Егор Олегович,** студент Воронежского государственного технического университета, Воронеж, Российская Федерация.

**Egor Olegovich Neretin,** Undergraduate Student, Voronezh State Technical University, Voronezh, the Russian Federation.

**Собенина Ольга Валерьевна,** кандидат технических наук, доцент Воронежского государственного технического университета, Воронеж, Российская Федерация.

**Olga Valerievna Sobenina,** Candidate of Technical Sciences, Associate Professor, Voronezh State Technical University, Voronezh, the Russian Federation.

*Статья поступила в редакцию 26.06.2023; одобрена после рецензирования 25.08.2023; принята к публикации 20.09.2023.*

*The article was submitted 26.06.2023; approved after reviewing 25.08.2023; accepted for publication 20.09.2023.*